Assignment\_RMarkDown

Snehasish

17/12/2020

R- Codes

# For comments  
# Shortcuts in R  
# ctrl+Enter ---> For execute of code  
# ctrl+ L -----> Clearing the console window  
# ctrl + 1 ------> shift cursor to R editor  
# ctrl + 2 -----> shift cursor to console window  
  
# creating simple objects and doing mathematical operations  
  
  
# Assigning value to a variable  
  
a = 5 # value 5 is assigned to variable a  
  
# Printing value assigned to a variable  
a # Printing the value of variable a

## [1] 5

a <- 6 # This is also one way of assigning value to a variable  
a

## [1] 6

scr = 'ggg' # Assigning character data to a variable . It can be done using either '' or ""  
scr

## [1] "ggg"

scrr = "dggg"  
scrr

## [1] "dggg"

print(class(a)) # Printing data type of variable a

## [1] "numeric"

print(class(scr))

## [1] "character"

print(class(scrr))

## [1] "character"

# class defines the data type of variable  
  
class(a) # Printing data type of variable a

## [1] "numeric"

class(scr)

## [1] "character"

class(scrr)

## [1] "character"

# Performing mathematical operations   
  
a - 8 # 8 is subtracted from value of variable a

## [1] -2

a

## [1] 6

b=TRUE  
b

## [1] TRUE

class(b)

## [1] "logical"

c = 4  
c

## [1] 4

a+c # Adding two variables

## [1] 10

# Using if condition  
if(b){  
 a+c  
}

## [1] 10

# Mathematical Operations  
sqrt(c)

## [1] 2

c^a

## [1] 4096

exp(c)

## [1] 54.59815

factorial(a)

## [1] 720

abs(a)

## [1] 6

cos(a)

## [1] 0.9601703

# Function  
  
# Divider is the name of function having two inputs  
divider = function(x,y) {  
 result = x/y  
 result  
}  
divider(2,4) # calling function for operation and providing two inputs for output

## [1] 0.5

multiplication = function(a,b) {  
 output = a\*b  
 output  
}  
multiplication(2,4)

## [1] 8

# concat and arrays  
  
fuc <- c(1,2,3,4,5) # Creating vector---using c (combine)  
fuc # Printing the array

## [1] 1 2 3 4 5

fuc + 10 # This will add 10 to all values individually

## [1] 11 12 13 14 15

fuc\*10

## [1] 10 20 30 40 50

k = c(1,2,3)  
k = c(a,b,c)  
k+1

## [1] 7 2 5

# Listing & deleting objects(variable)  
ls() # Listing all the variables created

## [1] "a" "b" "c" "divider"   
## [5] "fuc" "k" "multiplication" "scr"   
## [9] "scrr"

rm(scr) # Deleting the variable  
ls()

## [1] "a" "b" "c" "divider"   
## [5] "fuc" "k" "multiplication" "scrr"

rm (list = ls()) # Remove all the variables  
ls()

## character(0)

# Data types (Normal, Ordinal, Interval, Ratio)  
# But in system it is (Numeric, Character, Logical, Date, Vector)  
  
x= 10  
class(x)

## [1] "numeric"

# Numeric-- Integer and Decimal  
x = 10.54  
class(x)

## [1] "numeric"

x = 10L # to make it as a integer we have to add L  
class(x)

## [1] "integer"

# L - Integer  
is.integer(x) # checking whether the data type of x is an integer or not

## [1] TRUE

is.numeric(x) # # checking whether the data type of x is an numeric or not

## [1] TRUE

# Character--- Categorical Variable- Nominal  
s = "R studio"  
class(s)

## [1] "character"

# Characters----words/strings(Nominal), Classification(Gender- Male, Female)  
# Level of Classification- Factors----Involves Levels(Ordinals)  
  
# Logical  
a = TRUE  
class(a)

## [1] "logical"

is.logical(a)

## [1] TRUE

# Date---- 1 Jan 1970  
# POSIXct - Date plus Time  
date = as.Date("2012-06-28") # Format is yyyy-mm-dd  
date

## [1] "2012-06-28"

class(date)

## [1] "Date"

as.numeric(date)

## [1] 15519

date = as.POSIXct("2020-11-22 10:32:25") # Date + Time  
date

## [1] "2020-11-22 10:32:25 IST"

as.numeric(date)

## [1] 1606021345

# Vector   
# R is called as Vectorized language  
  
  
# vectors  
# A vector is collec tion of elements, all of same type.  
# A vector canot be of mixed type.  
# c- combine  
  
x = c(1,2,3,4,5,6,7,8,9,10) # Vector creation  
x

## [1] 1 2 3 4 5 6 7 8 9 10

# Arithmetic Operations on Vector  
x+1

## [1] 2 3 4 5 6 7 8 9 10 11

x-1

## [1] 0 1 2 3 4 5 6 7 8 9

c =x  
c-1

## [1] 0 1 2 3 4 5 6 7 8 9

c^2

## [1] 1 4 9 16 25 36 49 64 81 100

sqrt(c)

## [1] 1.000000 1.414214 1.732051 2.000000 2.236068 2.449490 2.645751 2.828427  
## [9] 3.000000 3.162278

sqrt(c^2)

## [1] 1 2 3 4 5 6 7 8 9 10

sqrt(c^4)

## [1] 1 4 9 16 25 36 49 64 81 100

# Vector creation  
  
a = 1:10 # This is also one method of creating vector . It will create sequence of nos from staring no. to ending no.  
a

## [1] 1 2 3 4 5 6 7 8 9 10

b = -5:4  
b

## [1] -5 -4 -3 -2 -1 0 1 2 3 4

a+b # Adding two vectors. Corresponding elements of one vector will be added to another.

## [1] -4 -2 0 2 4 6 8 10 12 14

a\*b

## [1] -5 -8 -9 -8 -5 0 7 16 27 40

length(a) # length of vector

## [1] 10

length(b)

## [1] 10

a

## [1] 1 2 3 4 5 6 7 8 9 10

a + c(1,2)

## [1] 2 4 4 6 6 8 8 10 10 12

a+c(1,2,3,4) # If Longer vector is not "multiple" of shorter vector, there will be warning

## Warning in a + c(1, 2, 3, 4): longer object length is not a multiple of shorter  
## object length

## [1] 2 4 6 8 6 8 10 12 10 12

# Vector comparisons  
  
a>5 # Checking whether vector values are greater than 5 or not

## [1] FALSE FALSE FALSE FALSE FALSE TRUE TRUE TRUE TRUE TRUE

a>0

## [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE

a>b # Checking whether vector values of a are greater than b or not

## [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE

a<b

## [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE

any(a<b) # Checking whether any value of vector a is less than value of vector b. If there it will return true else false.

## [1] FALSE

a

## [1] 1 2 3 4 5 6 7 8 9 10

b

## [1] -5 -4 -3 -2 -1 0 1 2 3 4

any(a>b)

## [1] TRUE

any(a<b)

## [1] FALSE

all(a<b) # Checking whether all value of vector a is less than value of vector b. If all the values satisfied then it will return true else false.

## [1] FALSE

all(a>b)

## [1] TRUE

c = c('cricket','football','basketball','hockey','athletics')  
nchar(c) # defines the no. of characters

## [1] 7 8 10 6 9

nchar(b)

## [1] 2 2 2 2 2 1 1 1 1 1

# Accessing individual elements in a vector  
  
c[1] # Accessing 1st element

## [1] "cricket"

c[0]

## character(0)

c[3]

## [1] "basketball"

c[1:2] # Accessing 1st to 2nd element

## [1] "cricket" "football"

c[1:4]

## [1] "cricket" "football" "basketball" "hockey"

c[c(1,4)] # Accessing 1st and 4th element

## [1] "cricket" "hockey"

# Assigning names to a vector   
d = c(q='one', w='two',e='three')  
d

## q w e   
## "one" "two" "three"

d[1]

## q   
## "one"

d[1:4]

## q w e <NA>   
## "one" "two" "three" NA

e = c(1:10, 20) # Vector creation  
e

## [1] 1 2 3 4 5 6 7 8 9 10 20

s = 1:3  
s

## [1] 1 2 3

# Assigning names after creation of vector  
names(s)= c('one','two','three')  
s

## one two three   
## 1 2 3

# Factors- Ordinal data  
  
q1 = c(c,'javellin','volleyball','shooting')  
length(q1)

## [1] 8

q1

## [1] "cricket" "football" "basketball" "hockey" "athletics"   
## [6] "javellin" "volleyball" "shooting"

q2 = c(q1,'hockey','cricket','badland')  
q2

## [1] "cricket" "football" "basketball" "hockey" "athletics"   
## [6] "javellin" "volleyball" "shooting" "hockey" "cricket"   
## [11] "badland"

q2\_F = as.factor(q2) # Select only unique value, removes dulplicate  
q2\_F

## [1] cricket football basketball hockey athletics javellin   
## [7] volleyball shooting hockey cricket badland   
## 9 Levels: athletics badland basketball cricket football hockey ... volleyball

class(q2)

## [1] "character"

as.numeric(q2\_F) # assigning unique integer to each value(based on alphabetical order)

## [1] 4 5 3 6 1 7 9 8 6 4 2

# Missing data in a vector  
  
x = c(1,2,3,NA,5,6,NA,8)  
x

## [1] 1 2 3 NA 5 6 NA 8

length(x)

## [1] 8

is.na(x)

## [1] FALSE FALSE FALSE TRUE FALSE FALSE TRUE FALSE

x = c(1,2,3,NA,5,6,NA,8,NULL,10) # NA means data not available there it will be counted in length but NULL means blank it will not be counted in length  
x

## [1] 1 2 3 NA 5 6 NA 8 10

class(x)

## [1] "numeric"

length(x)

## [1] 9

is.null(x)

## [1] FALSE

y = c('hockey',NA,'cricket')  
y

## [1] "hockey" NA "cricket"

class(y)

## [1] "character"

is.na(y)

## [1] FALSE TRUE FALSE

z = c(1,NULL,2)  
z

## [1] 1 2

is.null(z)

## [1] FALSE

# Data Structure  
# DATA frame  
# In Data Frame each individual column is a vector of same length. Each column can hold different types of data. In one column data type should be same.  
  
x = 10:1 # Vector  
y = -4:5  
z = c("Hockey","Football", "Cricket","vollleyball","xtx","gfygg","ggfgg","kkkkk","llll","oooo")  
# Creating Data frame from multiple vectors  
w = data.frame(x,y,z) # x,y,z will be three columns  
w

## x y z  
## 1 10 -4 Hockey  
## 2 9 -3 Football  
## 3 8 -2 Cricket  
## 4 7 -1 vollleyball  
## 5 6 0 xtx  
## 6 5 1 gfygg  
## 7 4 2 ggfgg  
## 8 3 3 kkkkk  
## 9 2 4 llll  
## 10 1 5 oooo

str(w) # structure of the data frame. It tells how many columns are there and their data type

## 'data.frame': 10 obs. of 3 variables:  
## $ x: int 10 9 8 7 6 5 4 3 2 1  
## $ y: int -4 -3 -2 -1 0 1 2 3 4 5  
## $ z: chr "Hockey" "Football" "Cricket" "vollleyball" ...

z = as.factor(z)  
w = data.frame(First=x, Second=y, Third=z) # Naming the columns  
w

## First Second Third  
## 1 10 -4 Hockey  
## 2 9 -3 Football  
## 3 8 -2 Cricket  
## 4 7 -1 vollleyball  
## 5 6 0 xtx  
## 6 5 1 gfygg  
## 7 4 2 ggfgg  
## 8 3 3 kkkkk  
## 9 2 4 llll  
## 10 1 5 oooo

# Checking the Dimension of Data Frame  
  
nrow(w) # No. of Rows

## [1] 10

ncol(w) # No. of Columns

## [1] 3

dim(w) # Rows \* Column

## [1] 10 3

names(w) # Names of columns

## [1] "First" "Second" "Third"

rownames(w) # Row Names

## [1] "1" "2" "3" "4" "5" "6" "7" "8" "9" "10"

names(w)[3] # Selecting individual column name

## [1] "Third"

# Head and Tail  
  
head(w) # Print First 6 rows

## First Second Third  
## 1 10 -4 Hockey  
## 2 9 -3 Football  
## 3 8 -2 Cricket  
## 4 7 -1 vollleyball  
## 5 6 0 xtx  
## 6 5 1 gfygg

head(w, n=7) # Print First 7 rows

## First Second Third  
## 1 10 -4 Hockey  
## 2 9 -3 Football  
## 3 8 -2 Cricket  
## 4 7 -1 vollleyball  
## 5 6 0 xtx  
## 6 5 1 gfygg  
## 7 4 2 ggfgg

tail(w) # Print last 6 rows

## First Second Third  
## 5 6 0 xtx  
## 6 5 1 gfygg  
## 7 4 2 ggfgg  
## 8 3 3 kkkkk  
## 9 2 4 llll  
## 10 1 5 oooo

class(w)

## [1] "data.frame"

# Accessing individual column  
w$Third

## [1] Hockey Football Cricket vollleyball xtx gfygg   
## [7] ggfgg kkkkk llll oooo   
## 10 Levels: Cricket Football gfygg ggfgg Hockey kkkkk llll oooo ... xtx

# Accessing Specific row and column  
w[3,2] # 3rd row and 2nd Column

## [1] -2

w[3,2:3] # 3rd Row and column 2 through 3

## Second Third  
## 3 -2 Cricket

w[c(3,5), 2]# Row 3&5 , Column 2;

## [1] -2 0

# Matrix  
  
A = matrix(1:10, nrow = 5) # Create a 5x2 matrix  
A

## [,1] [,2]  
## [1,] 1 6  
## [2,] 2 7  
## [3,] 3 8  
## [4,] 4 9  
## [5,] 5 10

B = matrix(21:30, nrow = 5)   
B

## [,1] [,2]  
## [1,] 21 26  
## [2,] 22 27  
## [3,] 23 28  
## [4,] 24 29  
## [5,] 25 30

C = matrix(21:40, nrow = 2) # Create a 2x10 matrix  
C

## [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]  
## [1,] 21 23 25 27 29 31 33 35 37 39  
## [2,] 22 24 26 28 30 32 34 36 38 40

A+B # Matrix Addition

## [,1] [,2]  
## [1,] 22 32  
## [2,] 24 34  
## [3,] 26 36  
## [4,] 28 38  
## [5,] 30 40

A\*B # A = 5\*2 B = 5\*2

## [,1] [,2]  
## [1,] 21 156  
## [2,] 44 189  
## [3,] 69 224  
## [4,] 96 261  
## [5,] 125 300

A==B # checking whether elements are equal

## [,1] [,2]  
## [1,] FALSE FALSE  
## [2,] FALSE FALSE  
## [3,] FALSE FALSE  
## [4,] FALSE FALSE  
## [5,] FALSE FALSE

# Matrix Multiplication  
A %\*% t(B) # A is 5x2. B is 5x2. B-transpose is 2x5

## [,1] [,2] [,3] [,4] [,5]  
## [1,] 177 184 191 198 205  
## [2,] 224 233 242 251 260  
## [3,] 271 282 293 304 315  
## [4,] 318 331 344 357 370  
## [5,] 365 380 395 410 425

# Naming the Columns and Rows   
colnames(A) # Printing columns names of matrix A

## NULL

rownames(A)

## NULL

colnames(A)= c("Left","Right") # Assigning column names  
rownames(A)= c("1st","2nd","3rd","4th","5th")  
colnames(B)

## NULL

rownames(B)

## NULL

colnames(B)= c("First","Second")  
rownames(B)= c("One","Two","Three","Four","Five")  
colnames(C)

## NULL

rownames(C)

## NULL

colnames(C) = LETTERS [1:10]  
rownames(C) = c("Top", "Bottom")  
  
dim(A)

## [1] 5 2

dim(C)

## [1] 2 10

t(A)

## 1st 2nd 3rd 4th 5th  
## Left 1 2 3 4 5  
## Right 6 7 8 9 10

A %\*% C

## A B C D E F G H I J  
## 1st 153 167 181 195 209 223 237 251 265 279  
## 2nd 196 214 232 250 268 286 304 322 340 358  
## 3rd 239 261 283 305 327 349 371 393 415 437  
## 4th 282 308 334 360 386 412 438 464 490 516  
## 5th 325 355 385 415 445 475 505 535 565 595

# Arrays  
# It is a multi-dimensional vector  
a1 = array(1:12, dim = c(2,3,2)) # First element is Row Index, Second Element is Column Index and third element is outer dimension  
a1

## , , 1  
##   
## [,1] [,2] [,3]  
## [1,] 1 3 5  
## [2,] 2 4 6  
##   
## , , 2  
##   
## [,1] [,2] [,3]  
## [1,] 7 9 11  
## [2,] 8 10 12

a1 = array(1:12, dim = c(2,3,1))  
a1

## , , 1  
##   
## [,1] [,2] [,3]  
## [1,] 1 3 5  
## [2,] 2 4 6

a1 = array(1:12, dim = c(2,6,2))  
a1

## , , 1  
##   
## [,1] [,2] [,3] [,4] [,5] [,6]  
## [1,] 1 3 5 7 9 11  
## [2,] 2 4 6 8 10 12  
##   
## , , 2  
##   
## [,1] [,2] [,3] [,4] [,5] [,6]  
## [1,] 1 3 5 7 9 11  
## [2,] 2 4 6 8 10 12

a1 = array(1:12, dim = c(2,6,1))  
a1

## , , 1  
##   
## [,1] [,2] [,3] [,4] [,5] [,6]  
## [1,] 1 3 5 7 9 11  
## [2,] 2 4 6 8 10 12

a1[,,1] # Accessing 1st outer Dimension

## [,1] [,2] [,3] [,4] [,5] [,6]  
## [1,] 1 3 5 7 9 11  
## [2,] 2 4 6 8 10 12

a1[1,2,] # Accessing 1st row 2nd column in both the dimensions

## [1] 3

a1 = array(1:12, dim = c(2,3,3)) # Since after 2nd dim all nos are taken therefore it will repeat as in 1st dimension  
a1

## , , 1  
##   
## [,1] [,2] [,3]  
## [1,] 1 3 5  
## [2,] 2 4 6  
##   
## , , 2  
##   
## [,1] [,2] [,3]  
## [1,] 7 9 11  
## [2,] 8 10 12  
##   
## , , 3  
##   
## [,1] [,2] [,3]  
## [1,] 1 3 5  
## [2,] 2 4 6

a1 = array(1:12, dim = c(2,3,2))  
a1

## , , 1  
##   
## [,1] [,2] [,3]  
## [1,] 1 3 5  
## [2,] 2 4 6  
##   
## , , 2  
##   
## [,1] [,2] [,3]  
## [1,] 7 9 11  
## [2,] 8 10 12

a1[1,,] # Accessing 1st row in both the dimension

## [,1] [,2]  
## [1,] 1 7  
## [2,] 3 9  
## [3,] 5 11

# List  
# It Stores any number of items of any type.  
a2 = list(1,2,3) # Creating 3 element list  
a2

## [[1]]  
## [1] 1  
##   
## [[2]]  
## [1] 2  
##   
## [[3]]  
## [1] 3

a3 = list(c(1,2,3)) # Creating single element list(a vector)  
a3

## [[1]]  
## [1] 1 2 3

a4 = list(c(1,2,3), 4:7) # Creating 2 element list-- 1st element a 3 element vector, 2nd element a 4 element vector  
a4

## [[1]]  
## [1] 1 2 3  
##   
## [[2]]  
## [1] 4 5 6 7

a5 = list(w,1:10) # Creating list--1st element a data frame, 2nd a vector of 10 elements  
a6 = list(w,a2,a3)  
  
#Naming List (similar to column name in data.frame)   
names(a5)= c("data.frame", "vector")  
names(a5)

## [1] "data.frame" "vector"

a5

## $data.frame  
## First Second Third  
## 1 10 -4 Hockey  
## 2 9 -3 Football  
## 3 8 -2 Cricket  
## 4 7 -1 vollleyball  
## 5 6 0 xtx  
## 6 5 1 gfygg  
## 7 4 2 ggfgg  
## 8 3 3 kkkkk  
## 9 2 4 llll  
## 10 1 5 oooo  
##   
## $vector  
## [1] 1 2 3 4 5 6 7 8 9 10

#Naming using "Name-Value" pair  
a6 = list(DataFrame = w, Vector = a2, vector1 = a3)  
names(a6)

## [1] "DataFrame" "Vector" "vector1"

a6

## $DataFrame  
## First Second Third  
## 1 10 -4 Hockey  
## 2 9 -3 Football  
## 3 8 -2 Cricket  
## 4 7 -1 vollleyball  
## 5 6 0 xtx  
## 6 5 1 gfygg  
## 7 4 2 ggfgg  
## 8 3 3 kkkkk  
## 9 2 4 llll  
## 10 1 5 oooo  
##   
## $Vector  
## $Vector[[1]]  
## [1] 1  
##   
## $Vector[[2]]  
## [1] 2  
##   
## $Vector[[3]]  
## [1] 3  
##   
##   
## $vector1  
## $vector1[[1]]  
## [1] 1 2 3

# Creating an empty list  
(emptylist = vector(mode="list", length =4))

## [[1]]  
## NULL  
##   
## [[2]]  
## NULL  
##   
## [[3]]  
## NULL  
##   
## [[4]]  
## NULL

# LENGTH OF LIST  
length(a6)

## [1] 3

# Accessing elements  
a6[[1]] # Accessing 1st element

## First Second Third  
## 1 10 -4 Hockey  
## 2 9 -3 Football  
## 3 8 -2 Cricket  
## 4 7 -1 vollleyball  
## 5 6 0 xtx  
## 6 5 1 gfygg  
## 7 4 2 ggfgg  
## 8 3 3 kkkkk  
## 9 2 4 llll  
## 10 1 5 oooo

a6[["DataFrame"]] # Accessing elements using names

## First Second Third  
## 1 10 -4 Hockey  
## 2 9 -3 Football  
## 3 8 -2 Cricket  
## 4 7 -1 vollleyball  
## 5 6 0 xtx  
## 6 5 1 gfygg  
## 7 4 2 ggfgg  
## 8 3 3 kkkkk  
## 9 2 4 llll  
## 10 1 5 oooo

a6[[1]]$Third # Accessing column with name from 1st elements

## [1] Hockey Football Cricket vollleyball xtx gfygg   
## [7] ggfgg kkkkk llll oooo   
## 10 Levels: Cricket Football gfygg ggfgg Hockey kkkkk llll oooo ... xtx

a6[[1]][,"Second"]

## [1] -4 -3 -2 -1 0 1 2 3 4 5

a6[[1]][,"Second", drop = FALSE]

## Second  
## 1 -4  
## 2 -3  
## 3 -2  
## 4 -1  
## 5 0  
## 6 1  
## 7 2  
## 8 3  
## 9 4  
## 10 5

# Loading data into R  
  
b1 = "http://www.jaredlander.com/data/Tomato%20First.csv" # Loading csv file from the path  
b2 = read.table(file=b1,header = TRUE, sep = ",") # Read the csv file loaded in b1. # header means 1st row it will consider it as header else it will assign own header name  
head(b2)

## Round Tomato Price Source Sweet Acid Color Texture Overall  
## 1 1 Simpson SM 3.99 Whole Foods 2.8 2.8 3.7 3.4 3.4  
## 2 1 Tuttorosso (blue) 2.99 Pioneer 3.3 2.8 3.4 3.0 2.9  
## 3 1 Tuttorosso (green) 0.99 Pioneer 2.8 2.6 3.3 2.8 2.9  
## 4 1 La Fede SM DOP 3.99 Shop Rite 2.6 2.8 3.0 2.3 2.8  
## 5 2 Cento SM DOP 5.49 D Agostino 3.3 3.1 2.9 2.8 3.1  
## 6 2 Cento Organic 4.99 D Agostino 3.2 2.9 2.9 3.1 2.9  
## Avg.of.Totals Total.of.Avg  
## 1 16.1 16.1  
## 2 15.3 15.3  
## 3 14.3 14.3  
## 4 13.4 13.4  
## 5 14.4 15.2  
## 6 15.5 15.1

# Reading Text Files  
Garden = read.table("C:/Users/Sneha/Downloads/R-Test.txt",header=TRUE,sep="")  
head(Garden)

## Name ID  
## 1 aaa 10  
## 2 bbb 20  
## 3 ccc 30  
## 4 ddd 40  
## 5 eee 50  
## 6 fff 60

#R Binary Files  
# save the tomato data.frame to Disk  
save(b2, file="E:\\R\\4-Data Structure\\R-Test.rdata")  
# remove tomato from memory  
rm(b2)  
# Check if it still exists  
#head(b2)  
# read it from the rdata file  
load("E:\\R\\4-Data Structure\\R-Test.rdata")  
head(b2)

## Round Tomato Price Source Sweet Acid Color Texture Overall  
## 1 1 Simpson SM 3.99 Whole Foods 2.8 2.8 3.7 3.4 3.4  
## 2 1 Tuttorosso (blue) 2.99 Pioneer 3.3 2.8 3.4 3.0 2.9  
## 3 1 Tuttorosso (green) 0.99 Pioneer 2.8 2.6 3.3 2.8 2.9  
## 4 1 La Fede SM DOP 3.99 Shop Rite 2.6 2.8 3.0 2.3 2.8  
## 5 2 Cento SM DOP 5.49 D Agostino 3.3 3.1 2.9 2.8 3.1  
## 6 2 Cento Organic 4.99 D Agostino 3.2 2.9 2.9 3.1 2.9  
## Avg.of.Totals Total.of.Avg  
## 1 16.1 16.1  
## 2 15.3 15.3  
## 3 14.3 14.3  
## 4 13.4 13.4  
## 5 14.4 15.2  
## 6 15.5 15.1

# Read data from anywhere in the Disk/Computer  
#myData = read.csv(file.choose()) # No working directory setup is needed.  
  
  
# Basic statistics  
  
# Generate a random sample of 100 numbers between 1 and 100  
x = sample(x=1:100, size = 100,replace = TRUE) # Duplicate values present  
x

## [1] 90 76 55 85 23 78 64 98 77 43 47 65 89 28 19 45 98 88 36 86 14 32 91 17 44  
## [26] 5 3 21 21 74 32 33 68 97 73 37 19 77 29 83 21 89 43 70 53 46 64 12 85 42  
## [51] 92 36 22 81 98 95 38 45 56 82 19 60 92 25 32 4 31 1 37 94 61 29 27 95 91  
## [76] 28 76 58 91 93 70 34 68 84 46 30 12 13 8 95 85 79 63 45 53 11 54 65 22 90

x = sample(x=1:100, size = 100,replace = FALSE) # Unique values  
x

## [1] 67 15 50 93 37 46 40 100 87 45 79 63 72 31 23 86 17 70  
## [19] 89 27 2 47 4 59 56 6 7 18 9 68 57 44 61 58 92 54  
## [37] 33 90 66 62 51 91 55 43 26 34 96 80 75 22 20 19 77 8  
## [55] 85 30 81 74 94 38 98 97 11 64 1 83 52 24 60 3 95 16  
## [73] 49 48 41 42 76 99 71 73 32 65 88 10 25 14 84 78 28 39  
## [91] 36 13 12 29 82 35 5 53 69 21

mean(x) # Calculating Mean

## [1] 50.5

y = x  
y = sample(x=1:100, size = 20, replace = FALSE)   
y

## [1] 38 79 74 24 72 30 61 9 70 5 62 93 80 83 40 17 96 73 87 32

mean(y)

## [1] 56.25

y = sample(x, size = 20, replace = FALSE)  
y

## [1] 16 86 87 42 21 94 95 19 47 45 13 57 43 20 65 5 36 91 78 24

mean(y)

## [1] 49.2

z=x  
z[sample(x=1:100, size = 20, replace = FALSE)] = NA # 20 values will be NA in a sample of 100.  
z

## [1] 67 15 50 NA 37 46 40 100 87 45 79 63 NA 31 23 86 17 70  
## [19] 89 NA NA NA 4 59 NA 6 7 18 9 68 57 44 61 NA 92 54  
## [37] 33 90 66 62 51 91 55 43 26 34 96 80 75 22 20 NA 77 8  
## [55] NA 30 NA 74 94 38 NA 97 11 64 1 83 52 NA NA NA 95 NA  
## [73] 49 NA 41 42 76 NA 71 73 32 NA 88 10 25 14 84 78 28 NA  
## [91] 36 13 12 29 82 35 5 53 NA 21

mean(z)

## [1] NA

mean(z, na.rm=TRUE) # To calculate mean of sample containing NA value. We have to remove NA from sample.

## [1] 49.8625

# Weighted means  
grades = c(10,20,30,40)  
weights = c(1/2,1/4,1/8,1/8)  
weighted.mean(x= grades, w= weights) # Weighted Mean

## [1] 18.75

#Variance  
var(y)

## [1] 927.5368

# Variance using formula  
sum((y- mean(y))^2)/ (length(y)-1)

## [1] 927.5368

# Standard Deviation  
sqrt(var(y))

## [1] 30.45549

sd(y)

## [1] 30.45549

sd(z)

## [1] NA

sd(z, na.rm= TRUE)

## [1] 28.76564

# Other Commonly Used Functions  
min(x) # Minimum value of sample

## [1] 1

max(x) # Maximum value of sample

## [1] 100

median(x) # Median of Sample

## [1] 50.5

min(z)

## [1] NA

min(z, na.rm=TRUE)

## [1] 1

# Summary Statistics  
summary(x) # Irt will give Min, Max, Mean, median 1st and 3rd Quantile.

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1.00 25.75 50.50 50.50 75.25 100.00

summary(y)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 5.00 20.75 44.00 49.20 80.00 95.00

# Quantiles  
quantile(y, probs = c(0.25, 0.75)) # Calculate 25th and 75th Quantile

## 25% 75%   
## 20.75 80.00

quantile(y, probs = c(0.1,0.25,0.5, 0.75,0.99))

## 10% 25% 50% 75% 99%   
## 15.70 20.75 44.00 80.00 94.81

quantile(z, probs = c(0.25, 0.75), na.rm = TRUE)

## 25% 75%   
## 25.75 75.25

# Package Installation  
# install.packages("ggplot2")  
library(ggplot2)  
  
  
# T-tests  
  
data(tips, package = "reshape2") # Loading Datasets from Package  
head(tips) # Printing 1st6 rows of Dataset

## total\_bill tip sex smoker day time size  
## 1 16.99 1.01 Female No Sun Dinner 2  
## 2 10.34 1.66 Male No Sun Dinner 3  
## 3 21.01 3.50 Male No Sun Dinner 3  
## 4 23.68 3.31 Male No Sun Dinner 2  
## 5 24.59 3.61 Female No Sun Dinner 4  
## 6 25.29 4.71 Male No Sun Dinner 4

str(tips) # Print the structure of Dataset

## 'data.frame': 244 obs. of 7 variables:  
## $ total\_bill: num 17 10.3 21 23.7 24.6 ...  
## $ tip : num 1.01 1.66 3.5 3.31 3.61 4.71 2 3.12 1.96 3.23 ...  
## $ sex : Factor w/ 2 levels "Female","Male": 1 2 2 2 1 2 2 2 2 2 ...  
## $ smoker : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...  
## $ day : Factor w/ 4 levels "Fri","Sat","Sun",..: 3 3 3 3 3 3 3 3 3 3 ...  
## $ time : Factor w/ 2 levels "Dinner","Lunch": 1 1 1 1 1 1 1 1 1 1 ...  
## $ size : int 2 3 3 2 4 4 2 4 2 2 ...

write.csv(tips, "E:/R/5-Statistics/tips.csv", row.names = FALSE) # Saving csv file into location. Without any arbitrary row names.  
  
# Selecting unique values from a column  
unique(tips$sex)

## [1] Female Male   
## Levels: Female Male

unique(tips$day)

## [1] Sun Sat Thur Fri   
## Levels: Fri Sat Sun Thur

#One Sample t-test - ONE GROUP [Two Tail. Ho:Mean = 2.5]  
t.test(tips$tip, alternative = "two.sided", mu=2.5)

##   
## One Sample t-test  
##   
## data: tips$tip  
## t = 5.6253, df = 243, p-value = 5.08e-08  
## alternative hypothesis: true mean is not equal to 2.5  
## 95 percent confidence interval:  
## 2.823799 3.172758  
## sample estimates:  
## mean of x   
## 2.998279

#One Sample t-test - Upper Tail. Ho:Mean LE 2.5  
t.test(tips$tip, alternative = "greater", mu=2.5)

##   
## One Sample t-test  
##   
## data: tips$tip  
## t = 5.6253, df = 243, p-value = 2.54e-08  
## alternative hypothesis: true mean is greater than 2.5  
## 95 percent confidence interval:  
## 2.852023 Inf  
## sample estimates:  
## mean of x   
## 2.998279

# Two Sample T-test - TWO GROUP  
t.test(tip ~ sex, data = tips, var.equal = TRUE)

##   
## Two Sample t-test  
##   
## data: tip by sex  
## t = -1.3879, df = 242, p-value = 0.1665  
## alternative hypothesis: true difference in means is not equal to 0  
## 95 percent confidence interval:  
## -0.6197558 0.1074167  
## sample estimates:  
## mean in group Female mean in group Male   
## 2.833448 3.089618

#Paired- Two-Sample T-Test   
# install.packages("UsingR")  
require(UsingR)

## Loading required package: UsingR

## Loading required package: MASS

## Loading required package: HistData

## Loading required package: Hmisc

## Loading required package: lattice

## Loading required package: survival

## Loading required package: Formula

##   
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:base':  
##   
## format.pval, units

##   
## Attaching package: 'UsingR'

## The following object is masked \_by\_ '.GlobalEnv':  
##   
## grades

## The following object is masked from 'package:survival':  
##   
## cancer

head(father.son)

## fheight sheight  
## 1 65.04851 59.77827  
## 2 63.25094 63.21404  
## 3 64.95532 63.34242  
## 4 65.75250 62.79238  
## 5 61.13723 64.28113  
## 6 63.02254 64.24221

write.csv(father.son, "E:\\R\\5-Statistics\\father\_son.csv", row.names = FALSE)  
  
#ANOVA - Comparing Multiple Samples  
str(tips)

## 'data.frame': 244 obs. of 7 variables:  
## $ total\_bill: num 17 10.3 21 23.7 24.6 ...  
## $ tip : num 1.01 1.66 3.5 3.31 3.61 4.71 2 3.12 1.96 3.23 ...  
## $ sex : Factor w/ 2 levels "Female","Male": 1 2 2 2 1 2 2 2 2 2 ...  
## $ smoker : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...  
## $ day : Factor w/ 4 levels "Fri","Sat","Sun",..: 3 3 3 3 3 3 3 3 3 3 ...  
## $ time : Factor w/ 2 levels "Dinner","Lunch": 1 1 1 1 1 1 1 1 1 1 ...  
## $ size : int 2 3 3 2 4 4 2 4 2 2 ...

tipAnova = aov(tip ~ day, tips) # Comparing different samples, i.e there are 4 days in tips therefore there will be 4 samples regarding that.  
summary(tipAnova)

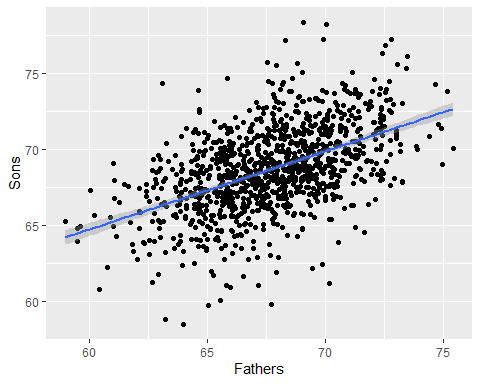
## Df Sum Sq Mean Sq F value Pr(>F)  
## day 3 9.5 3.175 1.672 0.174  
## Residuals 240 455.7 1.899

# Simple Linear Regression (SLR)  
# Using fathers' heights to predit sons' heights using SLR.  
# Fathers height as predictor(Indep - X) and   
# Son's height as the response /Target(Dep - Y)  
require(UsingR)  
require(ggplot2)  
head(father.son)

## fheight sheight  
## 1 65.04851 59.77827  
## 2 63.25094 63.21404  
## 3 64.95532 63.34242  
## 4 65.75250 62.79238  
## 5 61.13723 64.28113  
## 6 63.02254 64.24221

ggplot(father.son, aes(x=fheight, y=sheight))+geom\_point()+  
 geom\_smooth(method="lm")+labs(x="Fathers", y="Sons")

## `geom\_smooth()` using formula 'y ~ x'



heightsLM = lm(sheight ~ fheight, data = father.son)  
heightsLM

##   
## Call:  
## lm(formula = sheight ~ fheight, data = father.son)  
##   
## Coefficients:  
## (Intercept) fheight   
## 33.8866 0.5141

summary(heightsLM)

##   
## Call:  
## lm(formula = sheight ~ fheight, data = father.son)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -8.8772 -1.5144 -0.0079 1.6285 8.9685   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 33.88660 1.83235 18.49 <2e-16 \*\*\*  
## fheight 0.51409 0.02705 19.01 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 2.437 on 1076 degrees of freedom  
## Multiple R-squared: 0.2513, Adjusted R-squared: 0.2506   
## F-statistic: 361.2 on 1 and 1076 DF, p-value: < 2.2e-16